WhatsNext Vision Motors CRM Implementation Documentation

Prepared by: Yajnesh Ande

University: SRM-AP University

Project Overview

I developed a Salesforce CRM implementation for WhatsNext Vision Motors to enhance customer experience and streamline operational processes. The system automatically assigns orders to the nearest dealer based on customer location, prevents orders for out-of-stock vehicles, and includes automated workflows for order status updates. Key technical implementations include Apex triggers for stock validation, batch jobs for stock updates, and scheduled processes for automated order processing.

Objectives

My primary goal was to create an efficient ordering system that reduces errors and improves customer service. The objectives include implementing automatic dealer assignment based on customer location, preventing stock-related order issues, automating order status updates based on stock availability, and reducing administrative burden on staff through process automation.

Phase 1: Requirement Analysis & Planning

Understanding Business Requirements

I identified key business needs including automated dealer assignment, real-time stock validation during order placement, automated order status management, and scheduled email reminders for test drives. The system needed to handle bulk order processing and provide accurate stock information.

Defining Project Scope and Objectives

The scope focused on vehicle inventory management, dealer network administration, customer order processing, and test drive scheduling. I aimed to reduce order processing time, eliminate stock-related cancellations, and implement automated dealer assignment for all orders.

Design Data Model and Security Model

I designed a data model with six core objects: Vehicle, Vehicle_Dealer, Vehicle_Customer, Vehicle_Order, Vehicle_Test_Drive, and Vehicle_Service_Request. Each object includes proper relationships and security measures for data integrity.

Phase 2: Salesforce Development - Backend & Configurations

Setup Environment & DevOps Workflow

I created a developer org through the Salesforce Developer Program and established proper development practices with naming conventions and structured deployment approaches.

Customization of Objects, Fields, Validation Rules, Automation

I created six custom objects with the following key fields:

Vehicle__c: Vehicle_Name__c, Vehicle_Model__c (Picklist), Stock_Quantity__c, Price__c, Dealer__c (Lookup), Status__c (Picklist)

Vehicle_Dealer__c: Dealer_Name__c, Dealer_Location__c, Dealer_Code__c, Phone__c, Email__c

Vehicle_Customer__c: Customer_Name__c, Email__c, Phone__c, Address__c, Preferred_Vehicle_Type__c

Vehicle_Order__c: Customer__c (Lookup), Vehicle__c (Lookup), Order_Date__c, Status__c (Picklist)

Vehicle_Test_Drive__c: Customer__c (Lookup), Vehicle__c (Lookup), Test_Drive_Date__c, Status__c (Picklist)

Vehicle_Service_Request__c: Customer__c (Lookup), Vehicle__c (Lookup), Service_Date__c, Issue_Description__c, Status__c (Picklist)

I implemented two record-triggered flows:

Auto Assign Dealer Flow: Automatically assigns orders to the nearest dealer based on customer location when orders are created with "Pending" status.

Test Drive Reminder Flow: Sends automated email reminders to customers one day before their scheduled test drive.

Apex Classes, Triggers, Asynchronous Apex Classes

VehicleOrderTriggerHandler: Handles stock validation before order creation and updates vehicle stock when orders are confirmed.

VehicleOrderTrigger: Comprehensive trigger handling before and after events for insert and update operations.

VehicleOrderBatch: Batch job that processes pending orders, checking stock availability and updating order status from "Pending" to "Confirmed" when stock becomes available.

VehicleOrderBatchScheduler: Schedulable class that executes the batch job daily at midnight.

Phase 3: UI/UX Development & Customization

Lightning App Setup Through App Manager

I created the "WhatsNext Vision Motors" Lightning app including navigation items for all custom objects, Reports, and Dashboards. The app is configured for System Administrator profiles.

Page Layouts, Dynamic Forms

I designed intuitive page layouts for each custom object with proper field groupings and related lists for optimal user experience.

User Management

I established user management through System Administrator profile assignment with proper permissions for all objects and features.

Reports and Dashboards

I created essential reports for vehicle inventory, order status tracking, test drive schedules, and service request management to support business monitoring.

Phase 4: Data Migration, Testing & Security

Data Loading Process

I implemented data loading procedures using Salesforce's Data Import Wizard with proper data mapping and validation.

Field History Tracking, Duplicate Rules, Matching Rules

I configured field history tracking on critical fields and implemented duplicate rules to prevent data duplication and maintain data quality.

Profiles, Roles and Role Hierarchy, Permission Sets, Sharing Rules

I established a security model with appropriate profiles, permission sets, and sharing rules to ensure proper data access and security.

Testing Approach

I developed comprehensive testing including:

- Order creation with sufficient stock (Expected: Order confirmed, stock decremented)
- Order creation with insufficient stock (Expected: Error message, order prevented)

- Batch job execution (Expected: Pending orders updated to confirmed)
- Test drive reminder scheduling (Expected: Email sent one day before)
- Dealer assignment based on location (Expected: Nearest dealer assigned)

Phase 5: Deployment, Documentation & Maintenance

Deployment Strategy

I implemented deployment using change sets with proper testing and validation procedures to ensure system stability.

System Maintenance and Monitoring

I established monitoring procedures for batch job execution, flow performance, and regular maintenance tasks including data cleanup and performance optimization.

Documentation and Troubleshooting Approach

I developed comprehensive documentation covering system architecture, business processes, and technical implementations with systematic troubleshooting procedures.

Conclusion

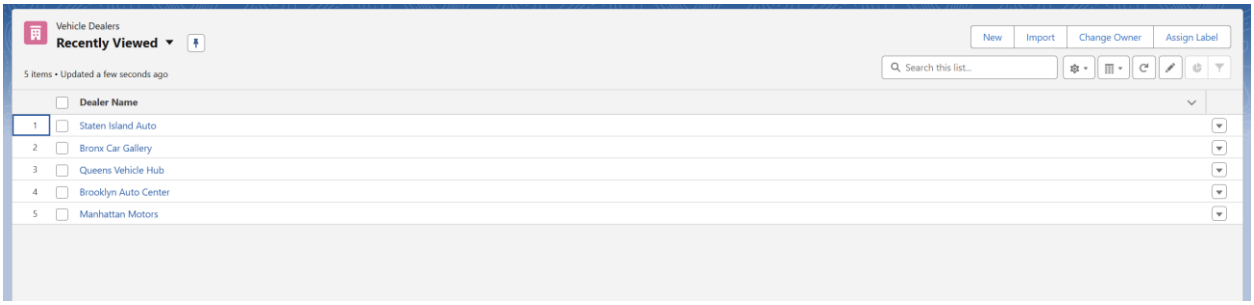
The WhatsNext Vision Motors CRM implementation successfully addresses key business challenges through automation and streamlined processes. The system enhances customer experience through automated dealer assignment, prevents stock-related issues through real-time validation, and improves operational efficiency through batch processing and scheduled automation.

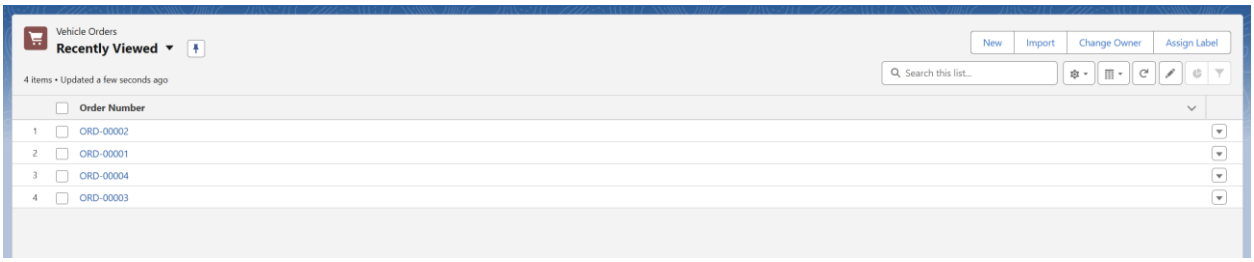
The project delivers measurable business value through improved order accuracy, reduced administrative burden, enhanced customer satisfaction, and streamlined operations. The technical architecture ensures system reliability and scalability for future business needs.

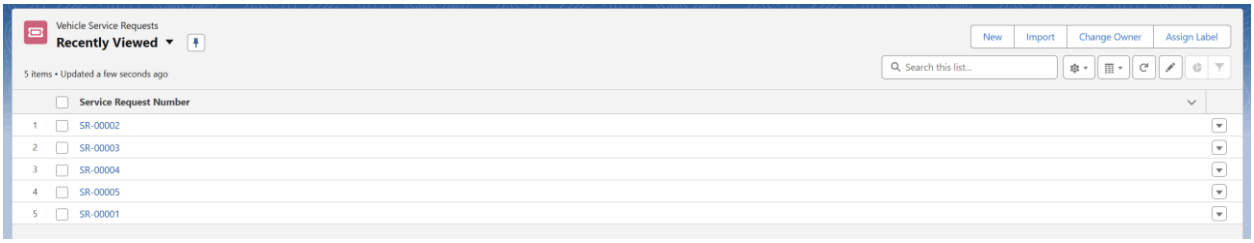
Future Enhancements

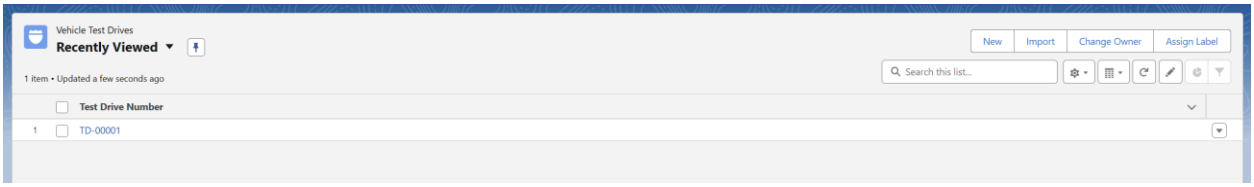
Future enhancements could include integration with external inventory systems, advanced analytics capabilities, customer self-service portals, and AI-driven vehicle recommendations based on customer preferences.

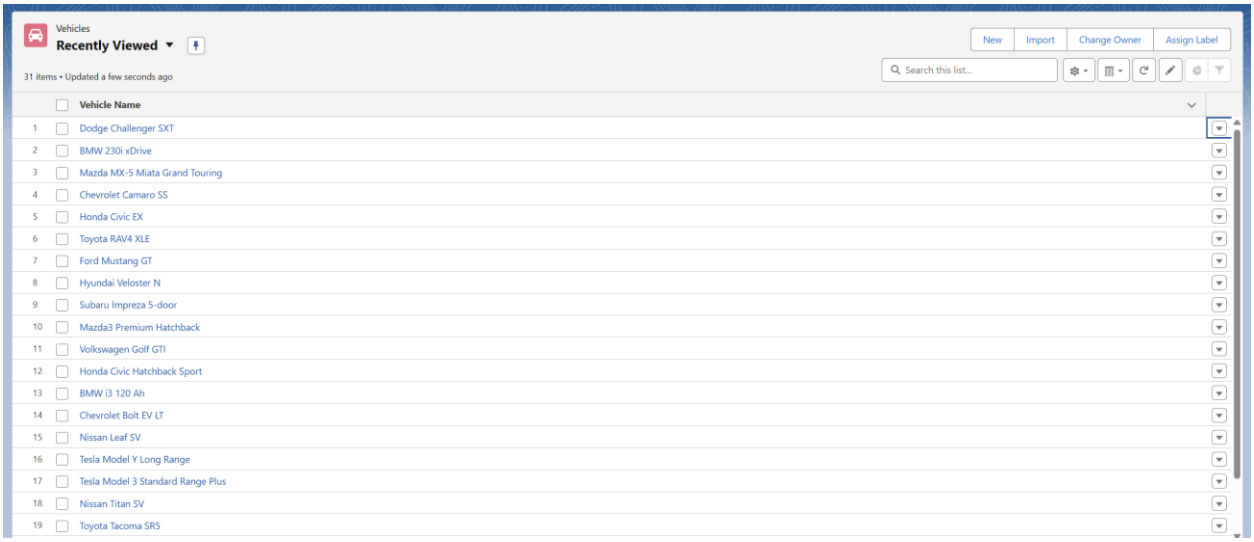
SCREENSHOTS:











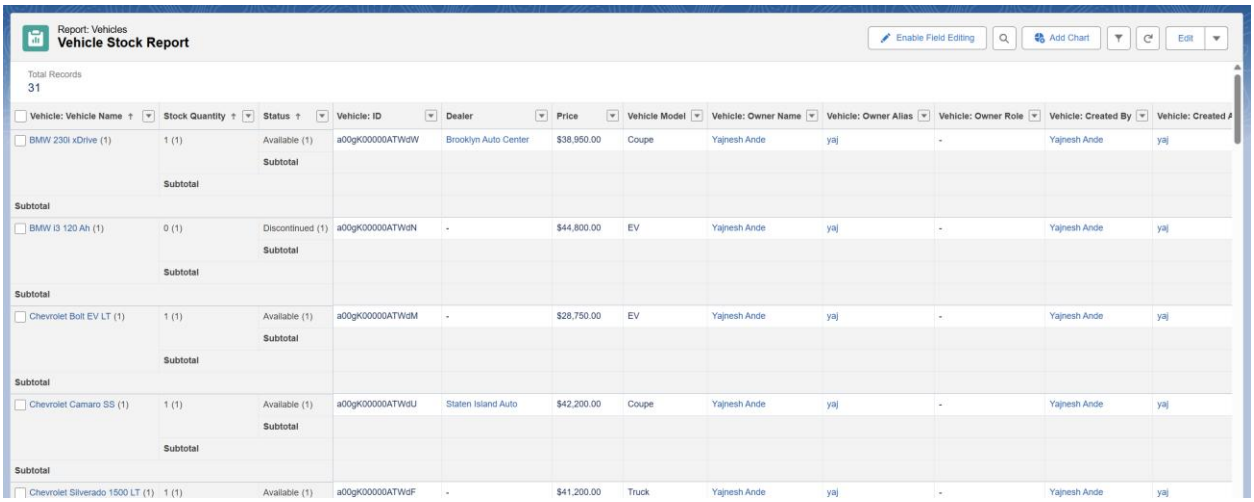
Report: Vehicle Orders

Order Status Report

Total Records

4

	Vehicle Order: Order Number	Customer	Vehicle	Status	Order Date
1	ORD-00002	Jennifer Garcia	Dodge Challenger SXT	Confirmed	7/14/2025
2	ORD-00004	Robert Wilson	BMW 230i xDrive	Delivered	7/25/2025
3	ORD-00001	Robert Wilson	Mazda MX-5 Miata Grand Touring	Confirmed	7/19/2025
4	ORD-00003	-	-	Pending	7/18/2025

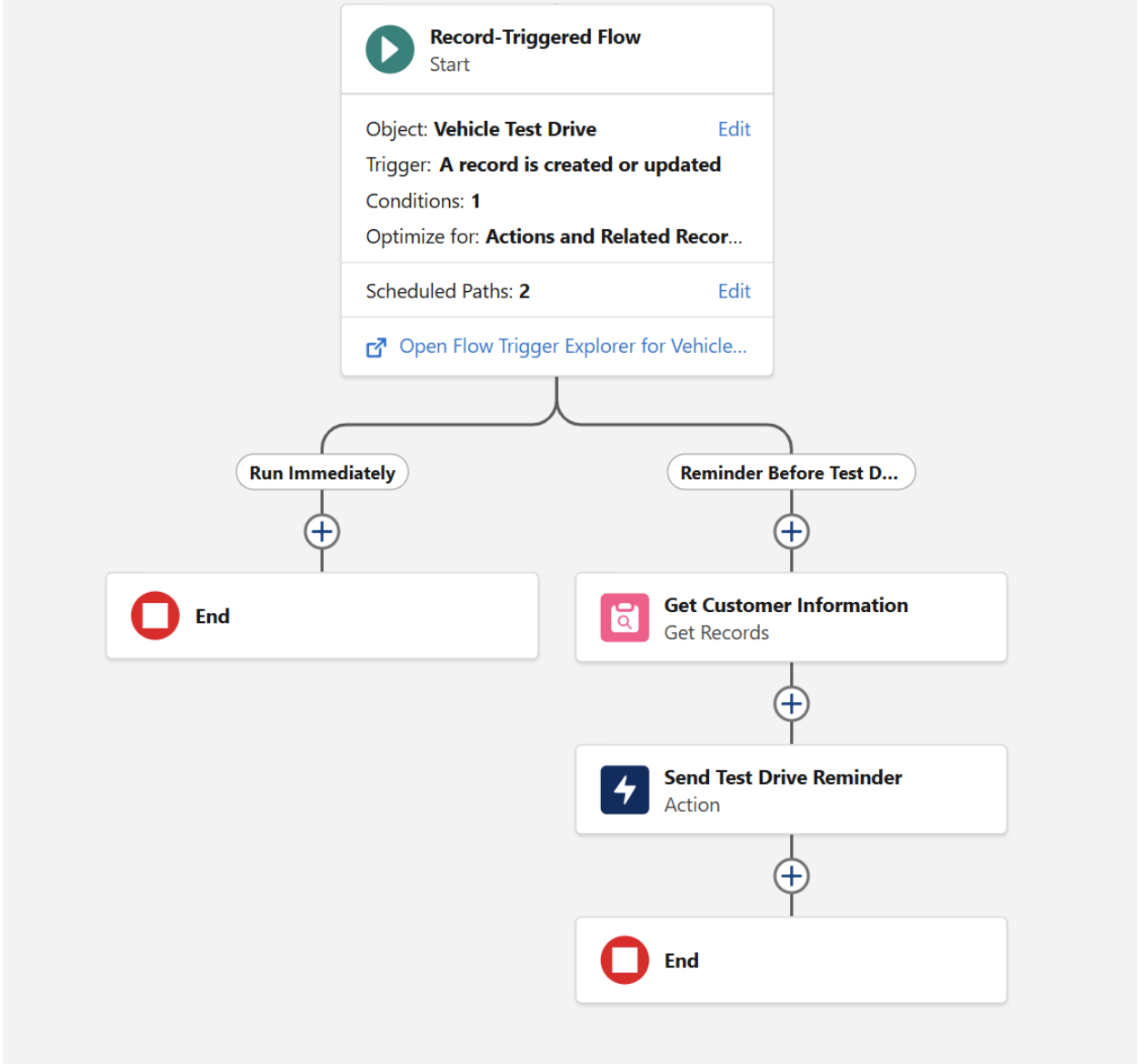


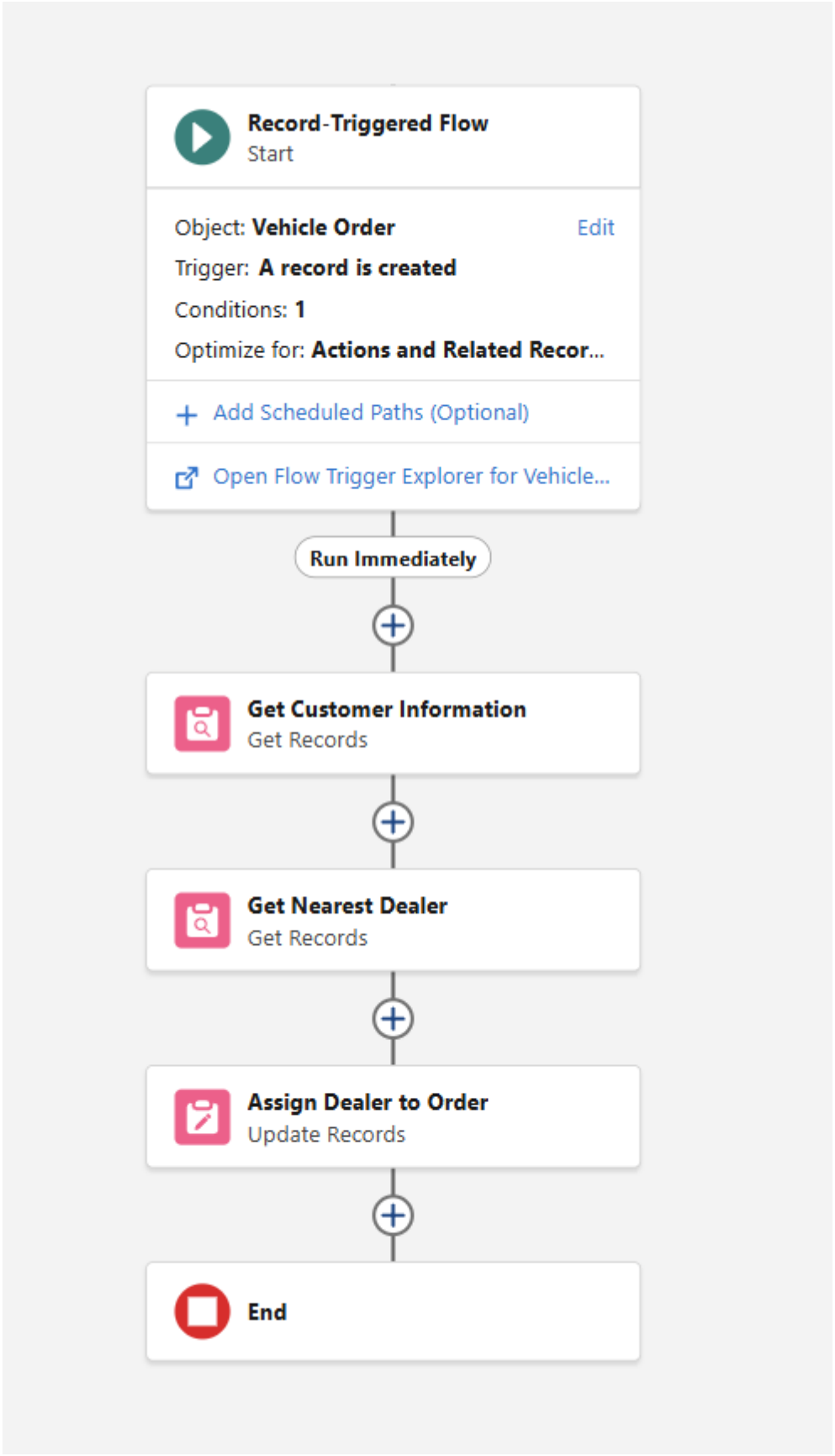
Report: Vehicle Test Drives

Test Drive Status Summary

Total Records
1

Status	Vehicle Test Drive: Test Drive Number	Customer	Vehicle
Scheduled (1)	TD-00001	Michael Brown	Mazda MX-5 Miata Grand Touring
Subtotal			
Total (1)			




Record-Triggered Flow
Start
Object: Vehicle Order
Edit
Trigger: A record is created
Conditions: 1
Optimize for: Actions and Related Recor...
Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Vehicle...
Run Immediately
Get Customer Information
Get Records
Get Nearest Dealer
Get Records
Assign Dealer to Order
Update Records
End

APEX CODE:

```
// Schedule Job Execution (Run Once)
String cronExp = '0 0 0 * * ?'; // Runs daily at 12:00 AM
System.schedule('Daily Vehicle Order Processing', cronExp, new VehicleOrderBatchScheduler());



// Scheduled Apex: VehicleOrderBatchScheduler
global class VehicleOrderBatchScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        VehicleOrderBatch batchJob = new VehicleOrderBatch();
        Database.executeBatch(batchJob, 50); // 50 is the batch size
    }
}

// Batch Apex: VehicleOrderBatch
global class VehicleOrderBatch implements Database.Batchable<sObject> {

    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([
            SELECT Id, Status__c, Vehicle__c
            FROM Vehicle_Order__c
            WHERE Status__c = 'Pending'
        ]);
    }

    global void execute(Database.BatchableContext bc, List<Vehicle_Order__c> orderList) {
```

```
Set<Id> vehicleIds = new Set<Id>();
for (Vehicle_Order__c order : orderList) {
    if (order.Vehicle__c != null) {
        vehicleIds.add(order.Vehicle__c);
    }
}

if (!vehicleIds.isEmpty()) {
    Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
        [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
    );

    List<Vehicle_Order__c> ordersToUpdate = new List<Vehicle_Order__c>();
    List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();

    for (Vehicle_Order__c order : orderList) {
        if (vehicleStockMap.containsKey(order.Vehicle__c)) {
            Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
            if (vehicle.Stock_Quantity__c > 0) {
                order.Status__c = 'Confirmed';
                vehicle.Stock_Quantity__c -= 1;
                ordersToUpdate.add(order);
                vehiclesToUpdate.add(vehicle);
            }
        }
    }
```

```
            if (!ordersToUpdate.isEmpty()) {
                update ordersToUpdate;
            }

            if (!vehiclesToUpdate.isEmpty()) {
                update vehiclesToUpdate;
            }
        }
    }

    global void finish(Database.BatchableContext bc) {
        System.debug('Vehicle order batch job completed.');
    }
}

// Apex Trigger: VehicleOrderTrigger
trigger VehicleOrderTrigger on Vehicle_Order__c (before insert, before update, after insert, after update) {
    VehicleOrderTriggerHandler.handleTrigger(Trigger.new, Trigger.oldMap, Trigger.isBefore, Trigger.isAfter, Trigger.isInsert, Trigger.isUpdate);
}

// Apex Class: VehicleOrderTriggerHandler
public class VehicleOrderTriggerHandler {
```

```
    public static void handleTrigger(List<Vehicle_Order__c> newOrders, Map<Id,
Vehicle_Order__c> oldOrders, Boolean isBefore, Boolean isAfter, Boolean isInsert, Boolean
isUpdate) {
        if (isBefore) {
            if (isInsert || isUpdate) {
                preventOrderIfOutOfStock(newOrders);
            }
        }
        if (isAfter) {
            if (isInsert || isUpdate) {
                updateStockOnOrderPlacement(newOrders);
            }
        }
    }

    private static void preventOrderIfOutOfStock(List<Vehicle_Order__c> orders) {
        Set<Id> vehicleIds = new Set<Id>();
        for (Vehicle_Order__c order : orders) {
            if (order.Vehicle__c != null) {
                vehicleIds.add(order.Vehicle__c);
            }
        }
        if (!vehicleIds.isEmpty()) {
            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
            );
            for (Vehicle_Order__c order : orders) {
```

```
            if (vehicleStockMap.containsKey(order.Vehicle__c)) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle.Stock_Quantity__c <= 0) {
                    order.addError('This vehicle is out of stock. Order cannot be placed.');
                }
            }
        }
    }
}

private static void updateStockOnOrderPlacement(List<Vehicle_Order__c> orders) {
    Set<Id> vehicleIds = new Set<Id>();
    for (Vehicle_Order__c order : orders) {
        if (order.Vehicle__c != null && order.Status__c == 'Confirmed') {
            vehicleIds.add(order.Vehicle__c);
        }
    }
    if (!vehicleIds.isEmpty()) {
        Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
            [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
        );
        List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();
        for (Vehicle_Order__c order : orders) {
            if (vehicleStockMap.containsKey(order.Vehicle__c)) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle.Stock_Quantity__c > 0) {
```

```
                vehicle.Stock_Quantity__c -= 1;
                vehiclesToUpdate.add(vehicle);
            }
        }
    }
    if (!vehiclesToUpdate.isEmpty()) {
        update vehiclesToUpdate;
    }
  }
}
```